CG2111A Engineering Principle and Practice II

Semester 2 2022/2023

"Alex to the Rescue"

Final Report

Team: B02-4B

Name	Student #	Sub-Team	Role
SIM JUSTIN	A0257926N	Software	Motor Control
CHUA ZHONG HENG	A0255320W	Hardware	Circuit design
ELLAWELA SUVEEN THINUSHA BANDARA	A0266706X	Software	Colour Sensor
KIRAN GEOFRAN THADIKARAN	A0258805W	Hardware	Alex Assembly

Preface

After nearly 9 months of EFFORT (Education, Fooling Around, Flunking, Overcoming challenges, Reevaluation, Triumph), we are finally able to see the fruits of our labour, Alex. Born as a maze-clearing robot, he comes as an upgrade to his prematurely born counterpart, MBot, with many additional capabilities. Following his birth, we celebrate by introducing him to the playground his younger sister once played in, to let him roam around freely (under adult supervision).

This project could not have been done without the assistance of the people around us. Many thanks to the professors of CG2111A for guiding us at every step along our path in realising Alex, especially Prof Boyd, the professor in charge of our lab, and Prof Collin, for standing in for Prof Boyd that one session he was unavailable. We would also like to thank the extremely helpful and friendly staff at the DSA lab for providing us with material aid, as well as our lab TAs Sean and Prince for all your help. And last but not least, our dear friends who helped us along the way. :)

With this, we proudly present to you our (pro)creation, our Alex Robot.

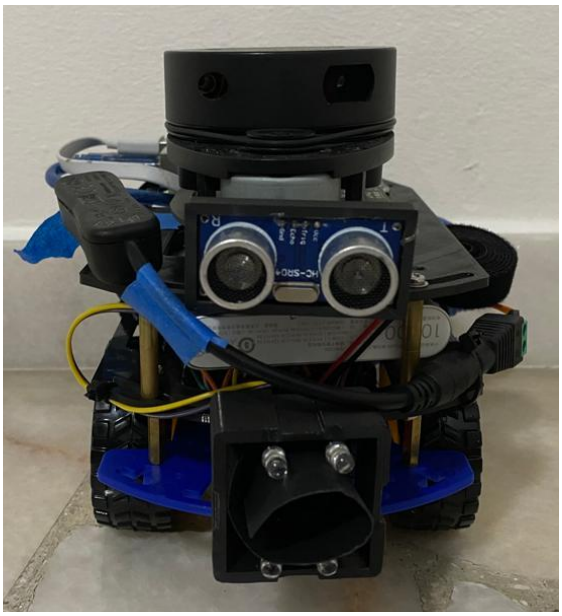


Figure I: Alex says Hi!

Table of Contents

Section 1: Introduction

As part of the CG2111A Final Project, we will be designing an “Alex” robot which aims to perform a simulated Search and Rescue task, serving as a platform to locate “survivors” in a “disaster site” in place of people in precarious environments.

1.1 Problem Statement

Alex is teleoperated with a laptop as the Ground Control Station (GCS) to navigate through a $3m^2$ area consisting of obstructions (a hump) and walls (forming 2-4 rooms) of 18 cm in height. Alex’s goal would be to, within a time limit, navigate from the start room to the end room with the help of an operator, who will receive an environment map from Alex during the run. Alex aims to perform this task while minimising object collisions and maintaining the fidelity of the generated environment map.

Additionally, specific objects demarking survivors (red or green) and dummy targets (any other colour) can be found, which the operator will have to identify from a LiDAR point cloud or environment map and send a request to Alex to traverse towards the target and determine its colour using the provided colour sensor, identifying whether it is a victim or a dummy target.

Section 2: Review of State of the Art

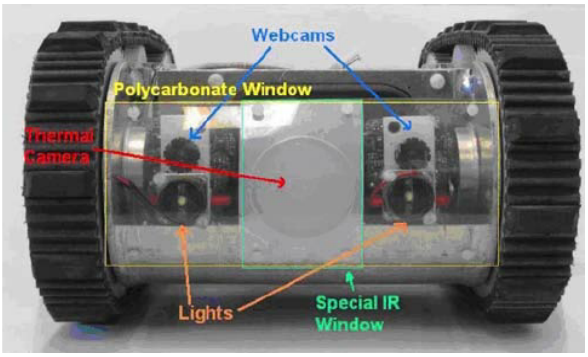
This is perhaps the end-stage goal for our Alex, perhaps he will be capable of becoming one of the following after he, and we, grows up.

2.1 Search and Rescue Robot with Tele-Operated Tether Docking System (RAPOSA)

We explore RAPOSA developed by IDMind and Institute for Systems and Robotics (Marques et al., 2007, 332) in our 1st case study.

2.1.1 Functionalities

RAPOSA is a tethered tracked robot designed for Urban Search and Rescue designed to operate in outdoor environments unsafe for human traversal, aiming to locate potential survivors of which information will be fed back to the remote human operator.



2.1.2 Hardware and Software

RAPOSA has a 2 segment chassis with track-based locomotion, allowing for traversal of rough terrain (Zhu et al., 2018, 1). Peripherals mounted on RAPOSA include thermal and optical cameras, gas, temperature, humidity sensors and LEDs, a

microphone and a loudspeaker. Actuation comprises a robotic arm controlled by multiple motors and sensors. To facilitate tethered operation, a cable is attached.

RAPOSA is teleoperated with a remote control unit and uses AI for navigation and object recognition, enabling it to make decisions based on sensor inputs. The control unit provides real-time monitoring and data management to the operator.

2.1.3 Strengths and Weaknesses

RAPOSA's tether eliminates its need for battery charging while its sensors and actuators enable it to perform necessary tasks during search and rescue tasks.

RAPOSA's tether however limits its mobility and range and the requirement for on-the-site operators may be problematic if trained operators are unavailable.

2.2 Airobotics Optimus

Our 2nd case study looks at Airobotics Optimus developed by ONDAS (ONDAS, n.d.).

2.1.1 Functionalities

Airobotics Optimus is a teleoperated UAV built for search and rescue operations targeted at disaster assessment and preliminary scene assessment.

2.1.2 Hardware and Software

Propulsion is achieved using a quadrotor setup and it is built with a variety of sensors including a gimbaled high-res camera and LIDAR. It is capable of wireless communication with a GCS.

AI improves its capabilities (i.e. autonomous flight, take-off, landing, collision avoidance, mapping). A proprietary OS allows for real-time flight control, autonomous operations, and remote monitoring along with storage and processing of sensor data in real time or in the future.

2.1.3 Strengths and Weaknesses

Airobotics Optimus features accurate sensors for precise mapping and identification of targets. Its AI capabilities allow for some degree of autonomous movement as well.

However, it is a costly system, and its flight time may limit its range.

Section 3: System Architecture

The GCS is the interface through which the operator handles Alex. Commands are sent from the GCS to Alex remotely via the Internet. We run ROS Melodic for Ubunto 18.04 to subscribe to LiDAR messages to perform SLAM, specifically Hector SLAM which does not require odometric data (Kohlbrecher et al., 2021).

The RasPi runs the master control program (MCP) for Alex. Data between the GCS and the Uno is compiled in this program, along with data from the LIDAR. The RasPi runs ROS Noetic for Debian 10 Buster, used to publish messages from the LiDAR module. Communication with the Uno is done serially via the USB B cable.

The Uno is a peripheral module used for controlling the motors and receiving data from the colour sensor and ultrasonic sensor. It interfaces with the RasPi, providing it with odometric, colour and distance data from the aforementioned sensors.

3.1 Alex UML Diagram

A UML deployment diagram for Alex is provided below. This highlights the connections between the various devices, notably the 3 key devices being the GCS (laptop), the RasPi and the Uno, and the flow of data between the various modules.

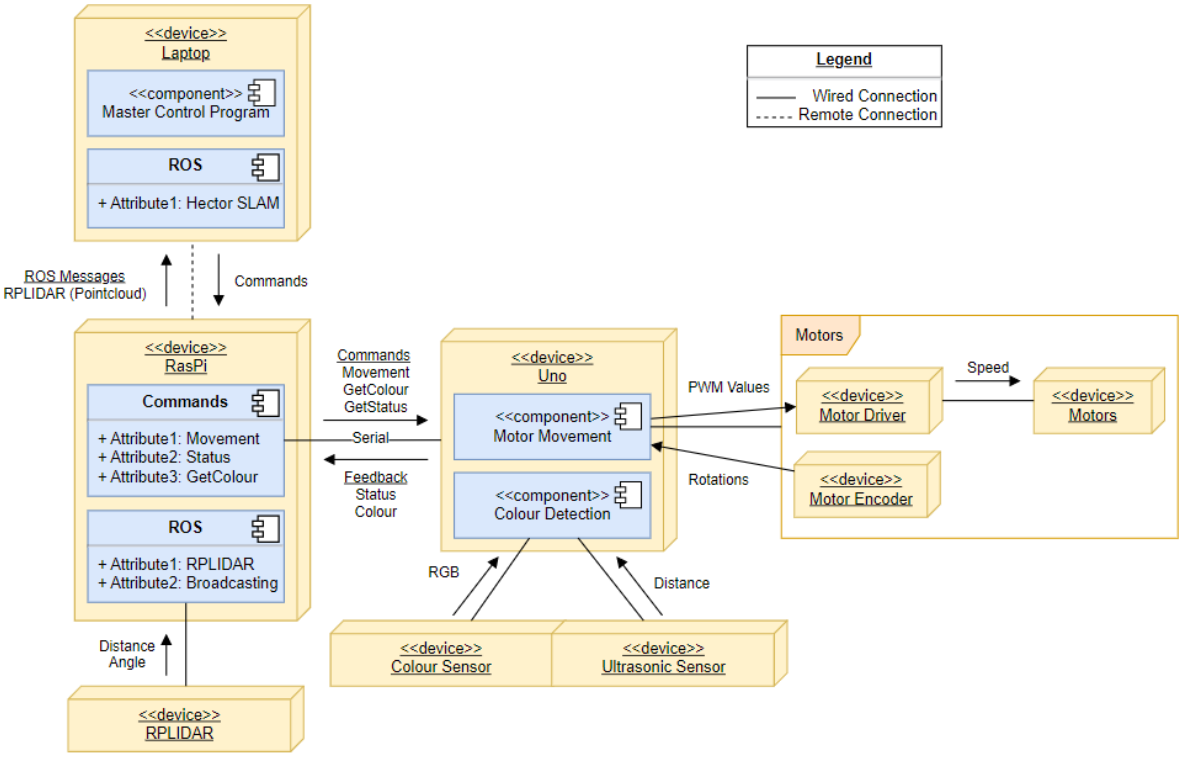


Figure II: UML Diagram for Alex System

Further information with regard to algorithm design and flow can be found in the subsequent sections on Firmware Design and Software Design.

Section 4: Hardware Design

4.1 Labelled Images of Alex

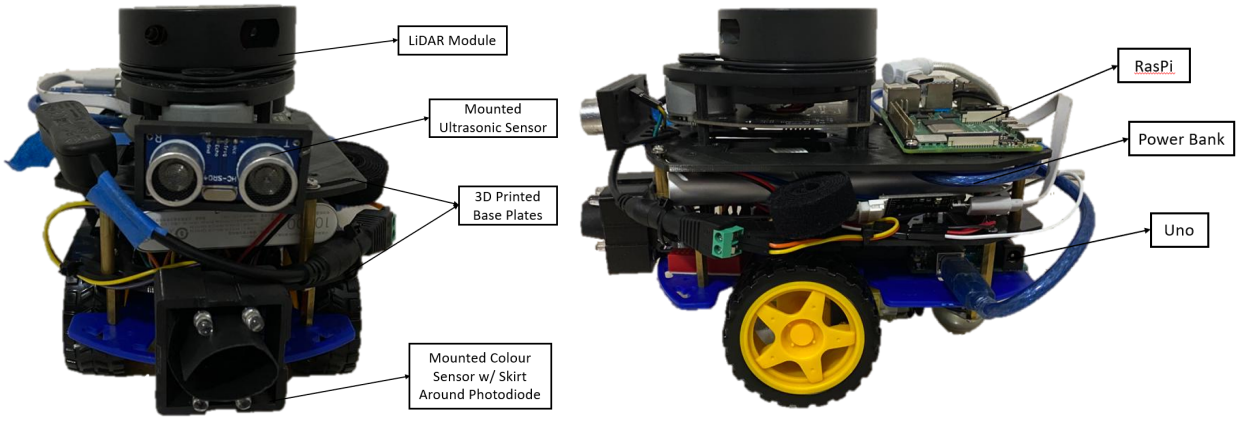


Figure III: Alex Front & Side Face

4.2 Hardware

Alex is designed with the Magician Robot Chassis and has a 2-wheel drive configuration. The motors are each connected to an encoder in order to determine the distance which each motor has moved. This helps us to achieve some form of odometry, albeit rather unreliable.

Power for Alex is split into power for the controllers and power for the motors. Controllers and their associated peripherals are powered using a power bank while the motors are powered via 4 rechargeable batteries.

Alex consists of an Arduino Uno microcontroller (henceforth referred to as Uno) and a Raspberry Pi 4 single-board computer (henceforth referred to as RasPi). The Uno is connected to a motor driver to control motor movement, along with the abovementioned encoders and a forward-pointed colour sensor, capable of returning the RGB frequencies of an object in front of it, and an ultrasonic sensor, capable of returning the distance between it and the nearest object in front of it. The RasPi is connected to the Uno via a USB B adaptor cable as well as a LiDAR module, the RPLIDAR-A1.

The schematic for the circuit centred around the Uno (since the RasPi is only connected to other devices via USB) is provided on the next page.

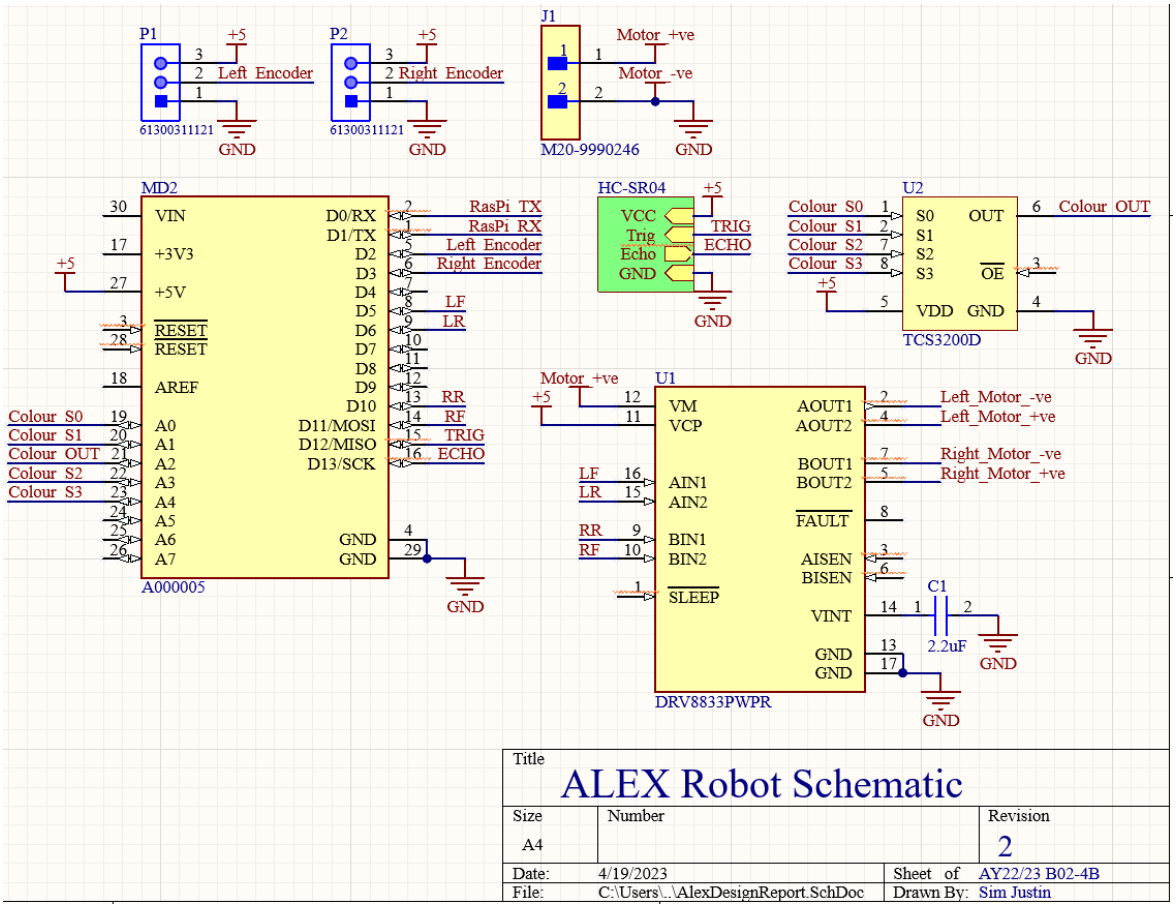


Figure IV: Schematic Diagram for Alex

4.3 Hardware Design Considerations

To aid our quest in accomplishing Alex's task, we have made a few hardware design considerations when building Alex. Given that Alex has a two-wheel drive configuration, we placed the LIDAR module on the axis of rotation such that there will be minimal translation of the module during turning, allowing for greater ease and accuracy of mapping the desired area.

Besides that, we also made other hardware inclusions as detailed below.

4.3.1 3D Printed Mounts

We 3D printed two layers in Alex (black middle layer and top layer) to aid us in our design, along with mounts for the colour sensor and the ultrasonic sensor on the front of Alex. 3D printing provides us with more agency in the placement of components aboard Alex (TWI, n.d.), allowing for greater ease of wire and device management. Images for these mounts can be found in Appendix A.

4.3.2 Black Paper Skirt Around Photodiodes

We included a skirt made of black paper around the photodiodes in the colour sensor to prevent light from the surrounding LEDs from affecting it. This change came after days of wondering why our colour sensor was not sensitive to changes in target colours and wondering why our raw values looked so different from other teams which implemented the skirt at an earlier stage.

4.3.3 Ultrasonic Sensor

We implemented an ultrasonic sensor at the front of the robot which is capable of returning the distance of the wall or an object in front of Alex to allow us to better estimate the room we have for manoeuvring. How this functionality is used will be further elaborated upon in the later section on Firmware Design.

4.4 Other Hardware Attempts

Due to the clarity of the initial ruleset, we trialled the implementation of a limit switch for the detection of the hump at first. However, we found a major issue during the trial run - our robot was unable to traverse over the hump as the switch got caught on the hump. This could potentially have been resolved by raising the switch slightly, but given the clarification of the rules during the trial run, we instead opted to remove it.

Figure V illustrates the mount we printed for the limit switch, which was supposed to be attached to the bottom of the lower base plate. The gap would be where the limit switch would have been attached and then electrically connected to the Uno.

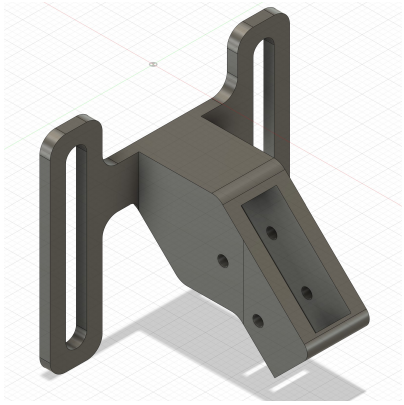


Figure V: Mount for Limit Switch

Section 5: Firmware Design

Our code base is stored on GitHub and can be found in the following public repository: https://github.com/1simjustin/CG2111A-Alex-Project. The repository has been organised into branches compiling the overall code, to be used on the GCS, to be used on Alex, miscellaneous files including headers and network files, and of course,

this very folder you are looking at right now. Unfortunately, we cannot claim to be Git masters by any measure so do pardon us for any blasphemies committed.

5.1 Algorithm on Uno

We use the following flowchart diagram to illustrate the high-level algorithm run on the Uno.

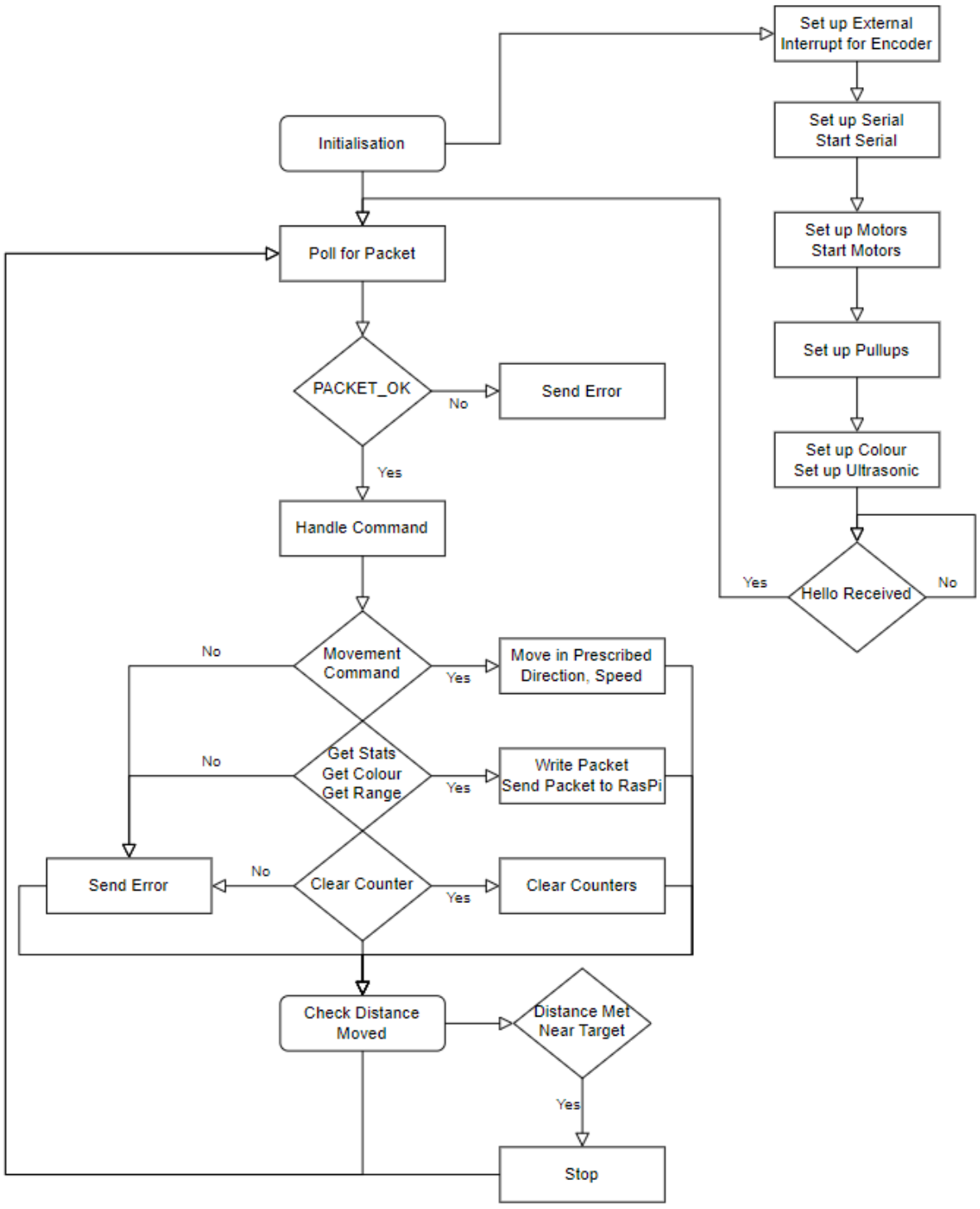


Figure VI: Flowchart for Uno

5.2 Communication Protocol

Communication between the GCS and the RasPi is done using TLS protocol, a secure means of data transfer over the internet using encryption (Cloudflare, n.d.). Our

implementation uses SHA256 encryption (Kumar, 2023), with ourselves as the Certifying Authority, preventing other users from injecting commands into our program.

Communication between the RasPi and the Uno is done serially, with a baud rate of 9600bps and a frame format of 8N1. On the Uno, this is achieved with bare-metal code by initialising UCSR0A to 0, UCSR0C to 0b110 and UBRR to 103 (for 9600 bps), then starting by setting UCSR0B to 0b11000, enabling TX and RX. This achieves serial communications in polling mode. Data is read by placing the bytes in UDR0 into a buffer when RXC0 in USCR0A is set, then deserializing it into a predefined packet struct mentioned below. Data is transmitted by serializing the packet into a buffer, and then writing the buffer to UDR0 sequentially when UDRE0 in UCSR0A is cleared.

5.2.1 Message Format

We did not make any meaningful changes to the format of the packets from earlier labs, only adding a few commands to allow Alex to perform his tasks. To save precious page space, we include the packet structure below and append the headers defining the respective packet contents in Appendix B.

```
typedef struct
{
    char packetType;
    char command;
    char dummy[2]; // Padding to make up 4 bytes
    char data[MAX_STR_LEN]; // String data
    uint32_t params[16];
} TPacket;
```

Figure VII: Packet Structure

We note that when attempting to employ a similar method to `handleStatus` for the sending of colour data, the `sslRead` function only returned the first element of params from the Uno to our GCS. This is an issue that we were unable to troubleshoot, hence we convert the colour data from the Uno to a string before forwarding it to the GCS as a message. We reference the following forum post for this problem (Jerebtsov, 2015).

5.3 Movement

Motors are set up for the locomotion of Alex using bare metal code as well. This is achieved by using all 3 counters TCNT0, TCNT1, and TCNT2. We initialise by setting TCCR0A, TCCR1A and TCCR2A to 0b1 for phase-correct PWM operation and enable the output pins by setting DDRB to 0b1100 and DDRD to 0b1100000. We then start the motors by setting TCCR0B, TCCR1B and TCCR2B to 0b11 to define the prescalar value of 64. Motor speed is then modified by editing OCR0A, OCR0B, OCR1B and OCR2A registers and direction by editing TCCR0A, TCCR1A and TCCR2A registers, thereby allowing us to traverse in all necessary directions.

We pass the desired duty cycle as a percentage into a function for control of the motor speeds. This function is also responsible for scaling down the speed for certain motors to ensure Alex does not veer.

Section 6: Software Design

6.1 Algorithm on RasPi & GCS

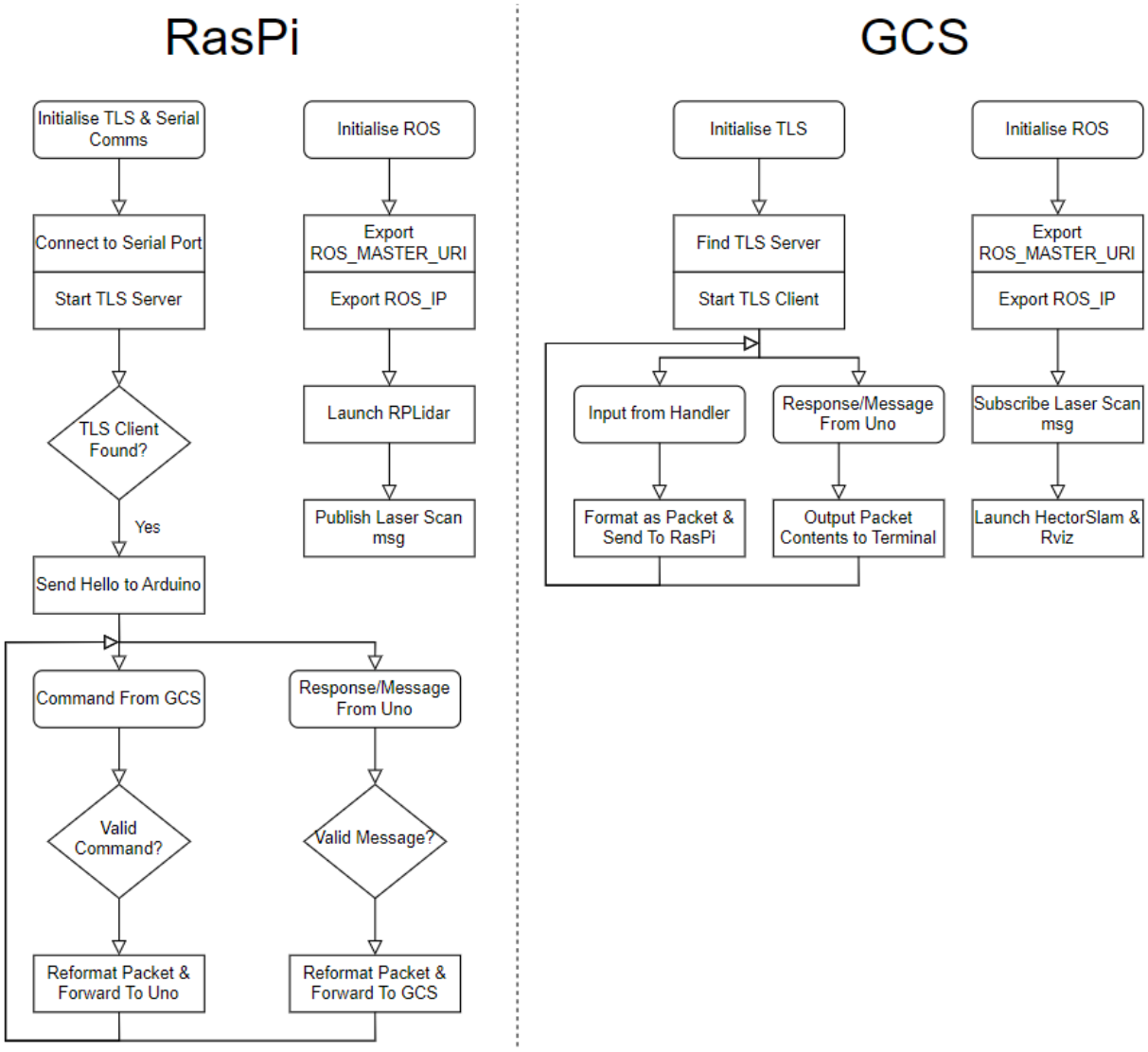


Figure VIII: Flowchart for RasPi & GCS

6.2 Alex Teleoperation

Teleoperation of Alex is done on the GCS is done on an on-command basis as opposed to a controller model, i.e. each command requires manual input followed by confirmation (command+enter) before the Alex executes the command. This was a

deliberate system architectural choice made in order to minimise the risk of unwanted excessive movement, especially considering the potential for latency. This provides us with the added benefit of having a wider range of commands which we can issue to Alex.

The commands which we can issue include movement and turning (WASD), cross hump (H), stop (E), clear stats (C), get stats (G), move to object and get colour (I) and get ultrasonic range (R). Additionally, we capitalise on the potential for added commands by using capitalisation of movement commands to dictate the distance which we want Alex to move by, i.e. WASD indicates to inch forward/backward/left/right while wasd would prompt us for the desired distance and speed to traverse. We fix the inch forward/backwards to be 5cm at 60% speed and the inch left/right to be 15 degrees at 60% speed.

6.3 Colour Detection

As part of the mission requirements, we are to identify multiple objects of which one will be red and another will be green, while the remaining can be of any colour. This task has to be performed without the use of a camera, forming the basis of the requirement for a means to detect the colour of an object using a colour sensor. To achieve this, the colour sensor connected to the Uno, on command, sends the raw values obtained from the colour sensor to the GCS for classification.

6.3.1 Colour Classification

Colour classification is performed on the GCS. This is done using a supervised learning (Song, 2022) complex neural network (Kavanaugh, 2020) consisting of roughly 86 billion neurons (Herculano-Houzel, 2012, 10661). In biological terms, many know of this as the human brain if it were not clear enough. This is achieved through astute observation by our esteemed teammate Suveen, who is able to identify the colour of a target from the raw colour sensor values following days of hard practice.

Colour Sensor Raw Values	R	B	G
Green	420-550	380-420	300-340
Red	150-200	510-540	690-780
White	0-150	0-150	0-150

Table I: Metrics for Determining Colour of Target (note RBG instead of RGB)

We initially attempted to implement a k-NN algorithm for colour classification (Pooja et al., 2021, 1041). However, we found that this algorithm is limited as it is ineffective in classifying data points outside the clusters it has been trained (Deepak,

2020), therefore being ineffectual in our use case where dummy targets can be of any colour.

6.3.2 Increasing Reading Accuracy

Alongside the hardware means of improving the accuracy of our colour sensor mentioned earlier, we have made a software modification to further improve the accuracy of readings. The distance of an object results in a change in the intensity of light (Richard, n.d.), which can distort the colour sensor readings. To avoid this, we included a movement command as part of COMMAND_GET_COLOR to move Alex to a fixed distance (7.5cm) from the object. To further increase accuracy, we continuously poll the distance returned by the ultrasonic sensor to determine when to stop rather than dictate a fixed distance to travel. This provides the added benefit of ensuring that we will not collide with our target. This can be viewed in the following video: https://youtu.be/V_IhP9xdZV0.

6.4 Use of ROS

Using ROS, we made a few deviations from what is expected from the earlier lab sessions. The key changes are highlighted below and can be seen in the following video: https://youtu.be/ImpBblPSx7o.

6.4.1 Remote Computation of Hector Slam and Rviz

Our ROS environment on the RasPi and the GCS was set up to allow for remote communication between the 2 devices. This allows us to offload the computationally intensive tasks of SLAM (Mustafa et al., 2017, 44) and visualisation of the generated map (Jinseop et al., 2022). With this, the only programs running on our RasPi are the MCP for communicating between the GCS and the Uno as well as the LiDAR module launch node.

6.4.2 Modelling Alex and the Environment in Rviz

As can be seen in the above video, we modelled the dimensions of Alex in ROS by creating 4 transformation frames outlining his 4 corners. This helps to ensure that we are aware of Alex's physical dimensions with respect to the maze to prevent collision with walls and other obstacles.

Section 7: Lessons Learnt - Conclusion

7.1 Lesson 1: Avoid Being Over-Ambitous

Being too ambitious in a project can lead to setting unrealistic expectations and goals that are difficult to achieve. This can lead to unnecessary stress as members try to work on features which are not necessary as feature creep takes over the goal. A case in point is our hopes to have a lot more functionalities as highlighted in our design report (i.e. return to base), which we were unable to accomplish in the end. It is thus

essential to set realistic goals which can be accomplished within a set deadline, keeping in mind what the desired end goal should be. This would then show an understanding of the task at hand, where working towards a minimum viable product would be the way to go instead of piling on increasing amounts of added features

7.2 Lesson 2: Changes in Client Needs

This project has taught us a key lesson on the dynamic nature of project and client needs and requirements. As can be seen from the frequent(?) updates to the mission ruleset, it is imperative to remain flexible in the face of such changes and adapt to the rising challenges. It is through the team's ability to respond to the changes in tasks and adapt to new requirements whereby the project can stay on track, creating a final product which can suit the needs of the customers. Communication with and between the team and the clients is key to ensuring the consistency of knowledge regarding the needs of the project across all parties. By embracing change and adapting to new challenges, our team was thus able to deliver a successful project meeting the challenge requirements of the final run.

7.3 Mistakes 1: Failure of Limit Switch

This failure mentioned above stems from our lack of testing under varying conditions. There are 2 ways to traverse the hump, along the grain and across the grain. Going along the grain would be the easier way to embark on the hump, and this was the only method we tested, neglecting the other method. This oversight led to our inability to overcome the hump during our trial run, instead causing Alex to get stuck on the hump. To prevent such errors from occurring in the future, the takeaway we gained from this mistake is the importance of a comprehensive testing plan that can cover all scenarios so as to ensure that we are able to, as early as possible, identify potential points of failure and make the necessary changes before final implementation.

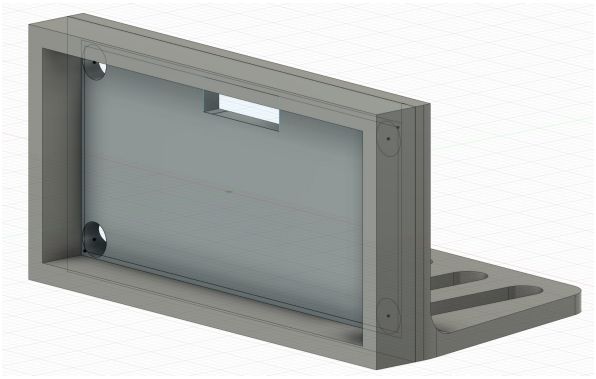
7.4 Mistakes 2: Overzealous Full Integration

In all our eagerness to have an assembled Alex as soon as possible, we failed to effectively prioritize our tasks, thereby neglecting the most critical stages of unit testing, resulting in significant time being wasted. This caused us to find many times when only one member was able to work on Alex at a given time. A notable example was when Justin was testing the motor control and Suveen wanted to test the implementation of the colour sensor and Zhong Heng wanted to redo the mounting for some components. This was a catastrophic failure and a massive waste of time for our team. To prevent this from occurring in the future, it is key all unit tests are conducted prior to final integration.

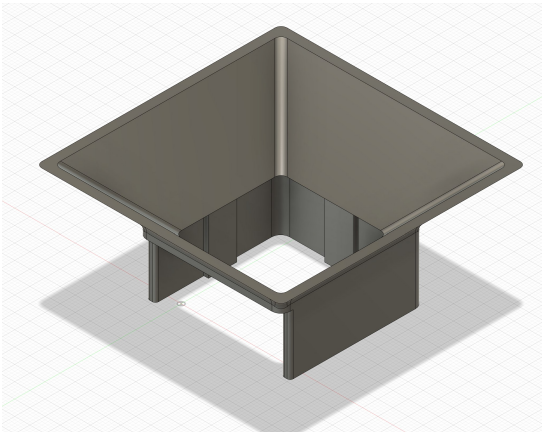
Appendix A

We include other 3D-printed designs in this section. It will be a slight spam of images but don't mind us.

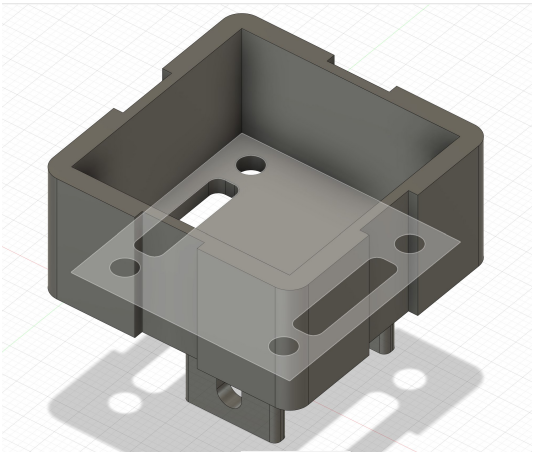
Ultrasonic Sensor Mount:



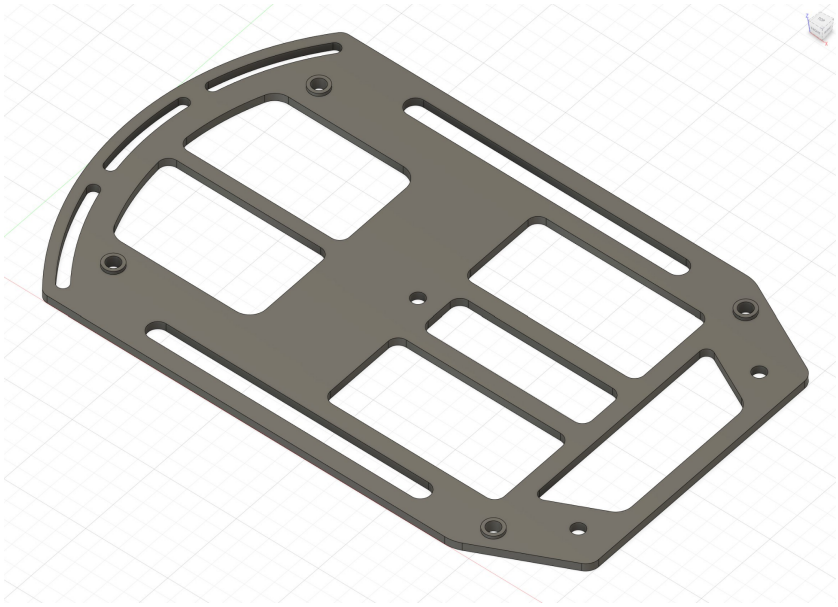
Colour Sensor Cover:



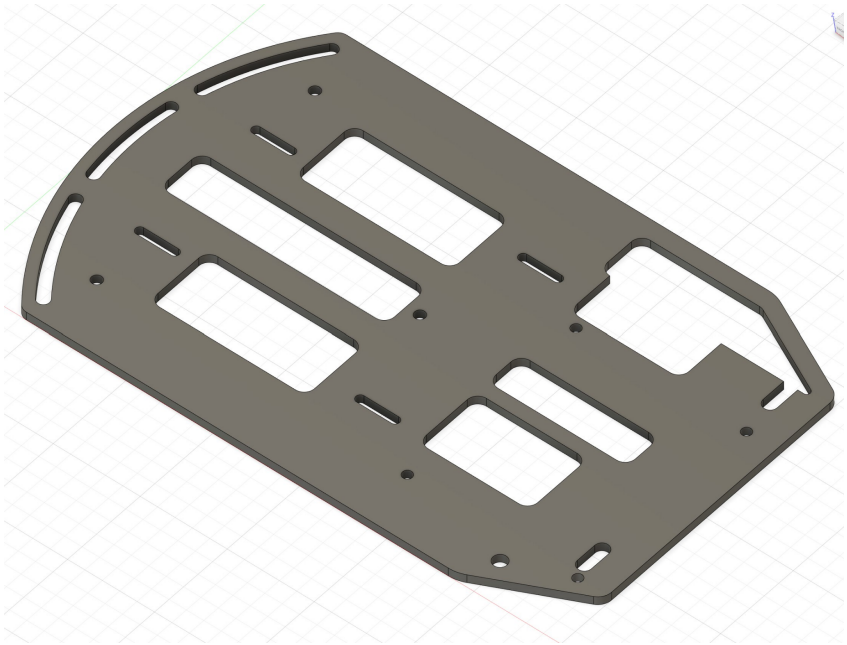
Colour Sensor Mount:



Middle Plate:



Top Plate:



Appendix B

RasPi-Uno Packet Contents

```
// Packet types
typedef enum
{
  PACKET_TYPE_COMMAND = 0,
  PACKET_TYPE_RESPONSE = 1,
  PACKET_TYPE_ERROR = 2,
  PACKET_TYPE_MESSAGE = 3,
  PACKET_TYPE_HELLO = 4,
} TPacketType;

// Response types. This goes into the command field
typedef enum
{
  RESP_OK = 0,
  RESP_STATUS= 1,
  RESP_BAD_PACKET = 2,
  RESP_BAD_CHECKSUM = 3,
  RESP_BAD_COMMAND = 4,
  RESP_BAD_RESPONSE = 5
} TResponseType;


// Commands
// For direction commands, param[0] = distance in cm to move
// param[1] = speed
typedef enum
{
  COMMAND_FORWARD = 0,
  COMMAND_REVERSE = 1,
  COMMAND_TURN_LEFT = 2,
  COMMAND_TURN_RIGHT = 3,
  COMMAND_STOP = 4,
  COMMAND_GET_STATS = 5,
  COMMAND_CLEAR_STATS = 6,
  COMMAND_GET_COLOR = 7,
  COMMAND_GET_RANGE = 8,
  COMMAND_HUMP = 9
} TCommandType;
```

GCS-RasPi Packet Contents

```
typedef enum
{
    NET_ERROR_PACKET=0,
    NET_STATUS_PACKET=1,
    NET_MESSAGE_PACKET=2,
    NET_COMMAND_PACKET=3
} TNetConstants;
```

References

Cloudflare. (n.d.). *What is Transport Layer Security? | TLS protocol*. Cloudflare. Retrieved April 19, 2023, from https://www.cloudflare.com/en-gb/learning/ssl/transport-layer-security-tls/

Deepak, J. (2020, July 17). *KNN: Failure cases, Limitations and Strategy to pick right K*. Level Up Coding. Retrieved April 18, 2023, from https://levelup.gitconnected.com/knn-failure-cases-limitations-and-strategy-to-pick-right-k-45de1b986428

Herculano-Houzel, S. (2012, April 12). The remarkable, yet not extraordinary, human brain as a scaled-up primate brain and its associated cost. *Proceedings of the National Academy of Sciences*, *109*(1), 10661 - 10668. PNAS. 10.1073/pnas.1201895109

Jerebtsov, I. (2015, July 15). *SslStream .read() returns with first byte once a packet*. Stack Overflow. Retrieved April 19, 2023, from https://stackoverflow.com/questions/31431747/sslstream-read-returns-with-first-byte-once-a-packet

Jinseop, J., Jun Yong, Y., Lee, H., Hatem, D., & Woosuk, S. (2022, September 18). Tutorial on High-Definition Map Generation for Automated Driving in Urban Environments. *Sensors*, *22*(18). mdpi. https://doi.org/10.3390/s22187056

Kavanaugh, B. (2020, April 22). *This is Your Brain… and This is Your Brain as a Neural Network*. Level Up Coding. Retrieved April 18, 2023, from

https://levelup.gitconnected.com/this-is-your-brain-this-is-your-brain-as-a-neural-network-fa50dd3e582d

Kohlbrecher, S., Kohlbrecher, S., Meyer, J., von Stryk, O., & Almeida, M. (2021, March 24). *hector_mapping*. ROS Wiki. Retrieved April 18, 2023, from http://wiki.ros.org/hector_mapping

Kumar, B. (2023, February 23). *What Is SHA-256 Algorithm: How it Works and Applications [2022 Edition]*. Simplilearn. Retrieved April 19, 2023, from https://www.simplilearn.com/tutorials/cyber-security-tutorial/sha-256-algorithm

Marques, C., Cristóvão, J., Alvito, P., Lima, P. U., Frazão, J., Ribeiro, I., & Ventura, R. (2007, June). A search and rescue robot with tele-operated tether docking system. *Industrial Robot: An International Journal*, *34*(3), 332-338. ResearchGate. 10.1108/01439910710749663

Mustafa, E., Ahmed, A., Islam, S., & Mahmoud, E. (2017). A critical comparison between Fast and Hector SLAM algorithms. *REST Journal on Emerging trends in Modelling and Manufacturing*, *3*(2), 44 - 49. restpublisher. ISSN: 2455-4537

ONDAS. (n.d.). *Optimus*. Airobotics. Retrieved March 23, 2023, from https://www.airoboticsdrones.com/optimus/

Pooja, K. S., Shreya, R. N., Yashika, B. C., Sree, L. M., & Rekha, B. N. (2021, August). Color Detection using K-Nearest Neighbors Classification Algorithm. *International Research Journal of Engineering and Technology*, *8*(8), 1041 - 1044. irjet.net. e-ISSN: 2395-0056

Richard, B. (n.d.). *How Does the Intensity of Light Change with Distance? | Science Project*. Science Buddies. Retrieved April 18, 2023, from

https://www.sciencebuddies.org/science-fair-projects/project-ideas/Elec_p028/electricity-electronics/measure-intensity-of-light

Song, C. (2022, April 13). *Supervised vs unsupervised learning*. Educative.io. Retrieved April 18, 2023, from https://www.educative.io/blog/supervised-vs-unsupervised-learning

TWI. (n.d.). *What are the Advantages and Disadvantages of 3D Printing?* TWI Global. Retrieved April 17, 2023, from https://www.twi-global.com/technical-knowledge/faqs/what-is-3d-printing/pros-and-cons

Zhu, Y., Kan, J., Li, W., & Kang, F. (2018, May 13). Strategies of traversing obstacles and the simulation for a forestry chassis. *International Journal of Advanced Robotic Systems*, *15*(3), 1 - 13. SAGE journals. 10.1177/1729881418773903